COBURG UNIVERSITY
of applied sciences and arts

Design Pattern: Chain-Of-Responsibility

Prof. Volkhard Pfeiffer

Coburg University of Applied Sciences and Arts

Gent, March 2019

Lecture Outlook

1. Chain-of-Responsibility (20 min)
2. Organisation (5 Min)
3. Exercise (50 min)
4. Break (10 min)
5. Presentation and discussion (30 min)
6. Summary (5 min)

References

1. Chain-Of-Responsibility

Intention
Problem
Structure
Example
Implementation
Summary

Intention

- **Decouple sender of a request to its receivers by giving more than one object a chance to handle the request**
- **Launch-and-Leave requests with a single processing pipeline that contains many possible handlers**

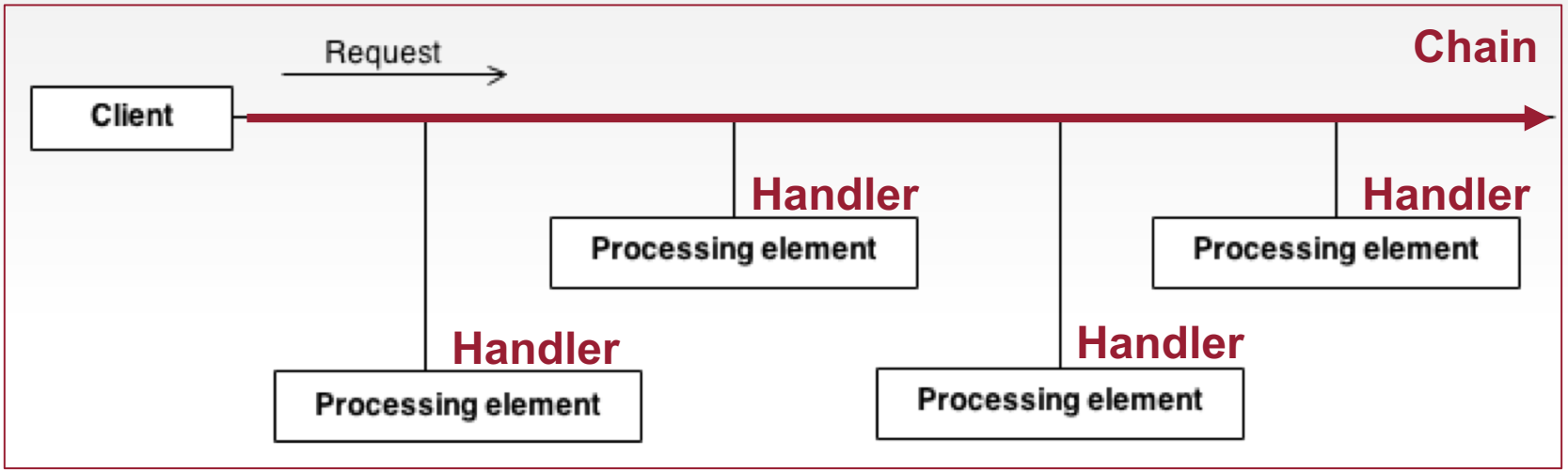
https://petroglobalnews.com/wp-content/uploads/2013/10/Screen-Shot-2013-10-30-at-10.21.23-AM.png

pipeline

processing stations

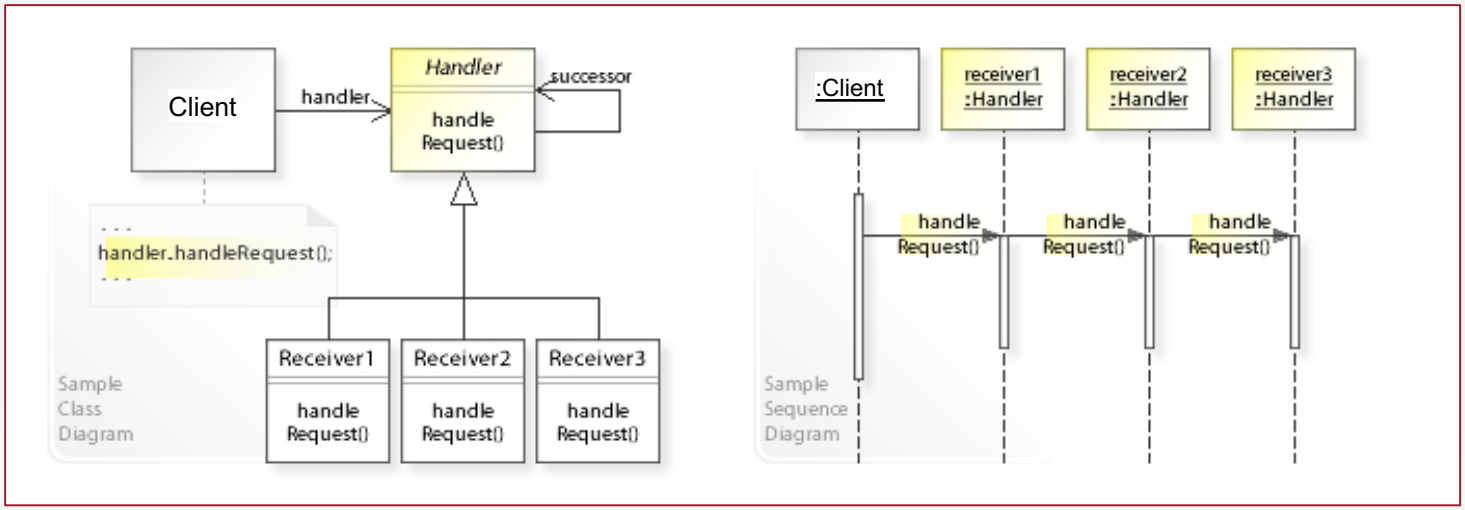
Problem

- variable number of „processing elements“ : Handler
- „stream“ of requests have to be handled
- process requests without hard-wiring handler relationships and precedence



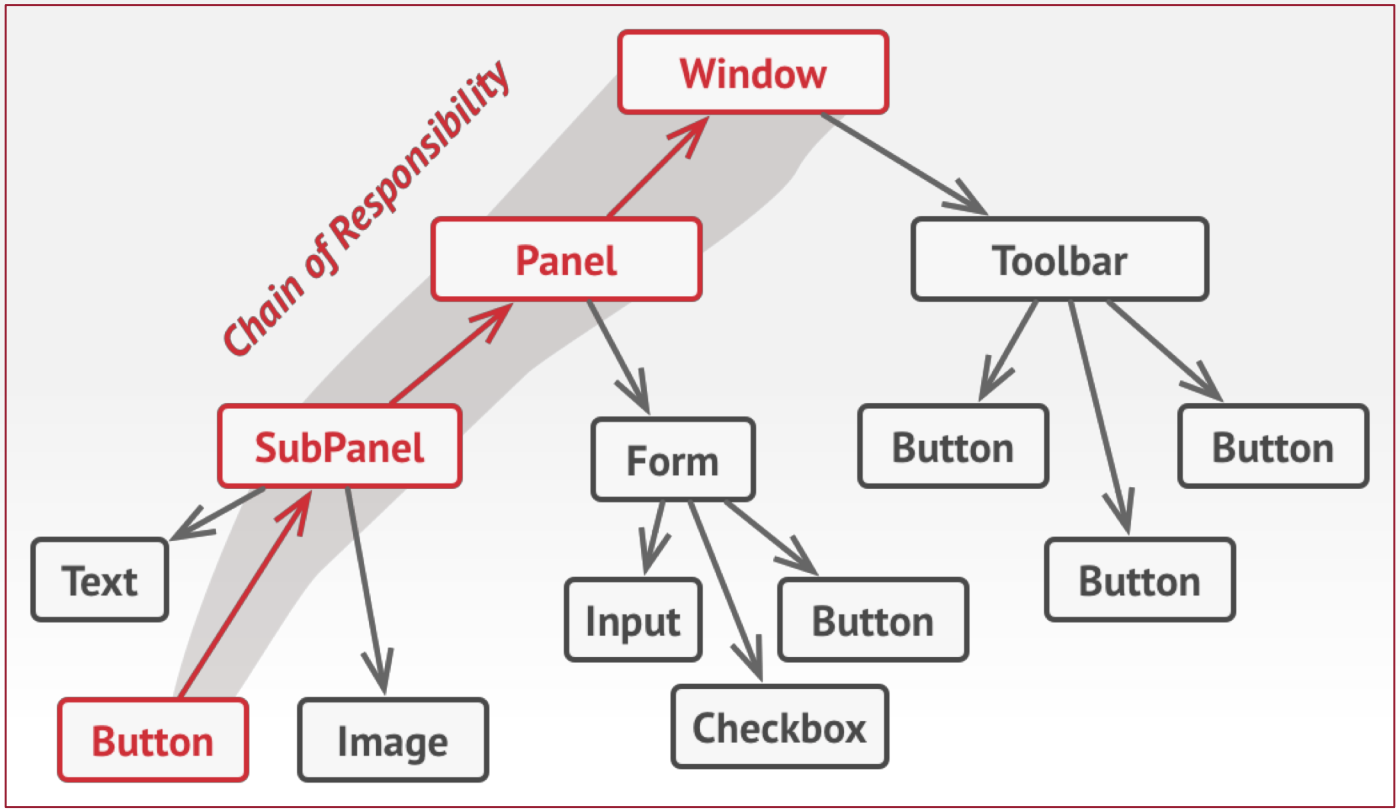
figur from [Explained Simply2019]

Structure



figur from https://en.wikipedia.org/wiki/Chain-of-responsibility_pattern

Example: Button Click Event propagation



figur from https://refactoring.guru/design-patterns/chain-of-responsibility

- starts with the button, goes along its containers, ends up main window
- event is processed by the first element that‘s capable of handling it

Implementation: All handlers have to implement the same interface

```
public abstract class CalculationHandler {

          protected CalculationHandler nextHandler;

          public void setNextCalculationHandler(CalculationHandler next) {
                    nextHandler = next;
          }

          public abstract void processCalculation(CalculationData request);
}
```

```
public class CalculationData {

          private int value;

          public CalculationData(int value) { … }

          public int  getValue() { … }

}
```

Handlers…

Attention: Handler code is incomplete: think about error handling

```
public class PositiveCalculationHandler extends CalculationHandler {

        @Override
        public void processCalculation(CalculationData request) {
            if (request.getValue() > 0)
                System.out.println("Positive values handled by
                                    PositiveCalculationHandler: " + request.getValue());
                // do something …
            else
                // process to the next handler
                nextHandler.processCalculation(request);
        }
}
```

```
public class NegativeCalculationHandler extends CalculationHandler {

        @Override
        public void processCalculation(CalculationData request) {
            if (request.getValue() < 0)
                System.out.println("Negative values handled by
                                    NegativeCalculationHandler: "+ request.getValue());
                // do something …
            else
                // process to the next handler
                nextHandler.processCalculation(request);
        }
}
```

….

Setup the chain and requests

```
public class Main {

     public static void main(String[] args) {

          // setup handler
          CalculationHandler c1 = new NegativeCalculationHandler();
          CalculationHandler c2 = new PositiveCalculationHandler();
          CalculationHandler c3 = new ZeroCalculationHandler();

          // setup the chain
          c1.setNextCalculationHandler(c2);
          c2.setNextCalculationHandler(c3);

          // requests
          c1.processCalculation(new CalculationData(-4));
          c1.processCalculation(new CalculationData(0));
          c1.processCalculation(new CalculationData(100));
      }
}
```

```
> Negative values handled by NegativeCalculationHandler: -4
> Zero values handled by ZeroCalculationHandler: 0
> Positive values handled by PositiveCalculationHandler: 100
```

2. – 4. Organisation, Exercise, Presentation and discussion

2. Organisation (5 min)

Form groups with 3 individuals. The following assignment has to be solved by each group.

3. Exercise (50 min)

Prepare a (short) presentation. The presentation should include the following aspects for your solution (*in your own words*):

- Java code solution
- Motivation/Applicability (when to use, when not to use)
- Participants (Classes and their roles)
- Collaboration (Chain explanation)
- Implementation Variants
  Different implementation variants do exist. Answer the following questions:
  - Which object(s) create(s) the chain?
  - How is the chain being executed?
  - How do you solve the “broken chain” problem?
- Pros/Cons/Consequences
- Optional: Related Pattern ((similar patterns and their differences/commonalities)

Outlook WrapUp

3. **Break** **(10 min)**

4. **Presentation and discussion** **(30 min)**

5. Summary

Pro‘s and Con‘s

- **Pro‘s**
  - *Single Responsibility Principle:* decouple classes that invoke operations from classes that perform operations.
  - *Open/Closed Principle:* You can introduce new handlers without breaking the existing client code.
  - Handler does not need to know the chain structure
  - enhance functional flexibility e.g. changing handlers or their order in the chain
- **Con‘s**
  - some requests may end up unhandled
  - system performance might be affected

Use the Chain of Responsibility pattern when

- … your program is expected to process different kinds of requests in various ways, but the exact types of requests and their sequences are unknown beforehand.
- … the request should be handled by one or more objects without specifying one receiver explicitly
- … the set of handlers and their order are supposed to change at runtime.

References

[Gamma+94] Gamma, E.; Helm, R.; Johnson, R.E.; Vlissides, J;
Design Patterns. Elements of reusable Object-Oriented Software
Prentice Hall 1994.

[Explained Simply2019] *Design Patterns – Explained Simply*
https://sourcemaking.com/design_patterns/chain_of_responsibility access 3/15/2019

[JournalDev2019] http://www.journaldev.com/1617/chain-of-responsibility-design-pattern-in-java additional Java ATM Code Example
access 3/15/2019

[OODesign2019] http://www.oodesign.com/chain-of-responsibility-pattern.html
access 3/15/2019

… and much more